**Индивидуальное задание**

**«Теория графов»**

***«Дискретная математика»***

|  |
| --- |
| **Выполнил:** |
| Студент 1-го курса механико-математического факультета  \_\_ Пьяных Андрей Сергеевич,  Вариант 4,  группа ПМИ-1,2. |

**Пермь 2021**

**Постановка задачи**

Определение двудольности графа

Входные данные.

В первой строке записано одно число n – количество вершин в

графе. Далее располагается матрица смежности графа (n строк по n чисел в каждой).

Выходные данные:

если граф является двудольным, то вывести в первой строке список

номеров вершин первой доли, во второй строке – список номеров вершин второй доли.

Если граф не является двудольным, то в первой строке вывести «NOT BIPARTITE», а

во второй строке вывести какой-нибудь цикл нечетной длины в виде последовательности

номеров вершин.

# Определение идеи алгоритма, выбор структур данных

Для решения задачи была выбрана система программирования Visual Studio 2017 и язык программирования C++.

Основные моменты:

* Выделение памяти
* Ввод
* Обход в глубину с окрашиванием пройденных вершин
* Вывод
* Освобождение памяти

Данные (графа) хранятся в виде структуры со следующими полями:

1. int n;
2. bool rez
3. int\*\* matrix
4. char\* color
5. int\* pred
6. clock\_t start
7. ifstream in("input.txt")
8. ofstream out("output.txt")
9. размер матрицы смежности
10. двудольный граф или нет
11. хранение матрицы смежности
12. массив для хранения цветов графа
13. массив для хранения предыдущего элемента при проходе в глубину
14. начало времени работы программы
15. переменная, для дальнейшого считывания данных из входного файла input.txt
16. переменная, для дальнейшого вывода данных из данных в файл output.txt

Поля 1,3 заполняются при вводе данных

Алгоритм выделения памяти:

1. Объявляем массивы int\*\* matrix; char\* color; int\* pred;
2. Считываем размер матрицы in >> n;
3. Выделяем памяти под наши массивы:

color = new char[n];

matrix = new int\* [n];

pred = new int[n];

1. Создаем двумерный массив и зануляем массив color

for (int i = 0; i < n; i++) {

matrix[i] = new int[n];

color[i] = 0;

}

Алгоритм ввода данных в двумерный массив:

Считываем однопоточно каждый символ из матрицы и записываем его в наш двумерный массив

for (int row = 0; row < n; row++)

for (int col = 0; col < n; col++)

in >> matrix[row][col];

Алгоритм обхода в глубину с окрашиванием пройденных вершин:

1. Задаем начальный цвет 1 вершине(будем использовать 2 цвета 1 и 2 и отсутствие цвета 0) color[0] = 1;
2. Вызываем фунцию обхода в глубину(dfs(matrix, n, 0, color, color[0],rez,out,pred)) и падаем ей на вход все нужные данные(матрицу смежности(matrix), количество вершин(n), номер текущей вершины(0,т.к. мы первой ее окрасили), массив цвета(color), текущий свет color[0], перменную двудольности(rez), переменную для вывода(out), масиив предыдущих вершин(pred))
3. Проходим по всей строке в поиске смежной вершины(обозначим ее S и обозначим текущую T), как только мы находим S, мы проверяем закрашена ли S.

* Если нет, то красим S в противоположный от T, и рекурсивно переходим в нее, помечаем, что T вершина предок S. И выполняем этот процесс, по всем вершинам или пока не найдем нечетный цикл.
* Если S закрашена, проверяем равен ли ее цвету T (now\_color), если да, то граф не двудольный и:
  + Присваиваем переменной двудольности false
  + Отмечаем, что T пердок S, для вывода нечетного цикла(search\_while)
  + Запускаем функцию search\_while, которой подаем на вход данные(

pred, значение S(i), n, значение T(number), out)

* + После завершения работы search\_while, завершаем работу всего алгоритма

void dfs(int\*\* matrix, int n, int number,char color[],char now\_color,bool& rez, ofstream& out,int pred[]) {

for (int i=0; i < n && rez; i++) {

if (matrix[i][number] == 1 && color[i] == 0) {

color[i] = invert(now\_color);

pred[i] = number;

dfs(matrix, n, i, color, color[i], rez,out,pred);

}

else

if (matrix[i][number] == 1 && color[i] == now\_color) {

rez = false;

pred[i] = number;

search\_while(pred, i, n, number,out);

return;

};

}

return;

}

Алгоритм инвертирования:

1. Если цвет равен 1, то присваиваем 2.
2. В противном случае, присваиваем 1.

char invert(char ch) {

if (ch == 1) return 2;

return 1;

}

Алоритм вывода цикла нечетной длинны:

Описываем условия выхода из рекурсии(if (t == begin))

Т.к. на вход подана одна из вершин цикла(begin), идем по ее предкам рекурсией, сравнивая на каждом этапе, текущую вершину t с begin

* Как только мы вернулись в begin, выводим сообщение о недвудольности и номер текущей вершины

out << "NOT BIPARTITE" << endl << t+1 << " ";

* На каждом этапе выводим номер текущей вершины и проходим дальше

search\_while(pred, begin, n, pred[t], out);

out << t + 1 << " ";

void search\_while(int pred[], int begin, int n, int t, ofstream& out) {

if (t == begin) {

out << "NOT BIPARTITE" << endl << t+1 << " ";

}

else {

search\_while(pred, begin, n, pred[t], out);

out << t + 1 << " ";

}

}

Алоритм вывода, если граф двудольный:

1. Проверяем переменную rez(которая выдает true,если граф двудолтный или false,когда нет). Если rez==true запускаем функцию вывода

if (rez) output(color, n, out);

1. Объявляем переменные для вывода данных string str1, str2;
2. Проходимся по color для распределения данных по строкам

string str1, str2;

for (int i = 0; i < n; i++)

if (color[i] == 1) str1 += to\_string(i + 1) + " ";

else

str2 += to\_string(i + 1) + " ";

1. Удаляем лишние пробелы из str1 и str2

str1.erase(str1.length() - 1, 1);

str2.erase(str2.length() - 1, 1);

1. Выводим данные

out << str1 << endl << str2;

void output(char color[], int n, ofstream& out) {

string str1, str2;

for (int i = 0; i < n; i++)

if (color[i] == 1) str1 += to\_string(i + 1) + " ";

else

str2 += to\_string(i + 1) + " ";

str1.erase(str1.length() - 1, 1);

str2.erase(str2.length() - 1, 1);

out << str1 << endl << str2;

}

Алгоритм освобождения памяти:

1. Удаляем массивы color,matrix,pred
2. Закрываем файлы с которыми работали

delete[] color;

for (int i = 0; i < n; i++) delete[] matrix[i];

delete[] matrix;

delete[] pred;

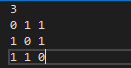
in.close();

out.close();

**Тестирование программы**

**Тест 1: (0,001sec)**

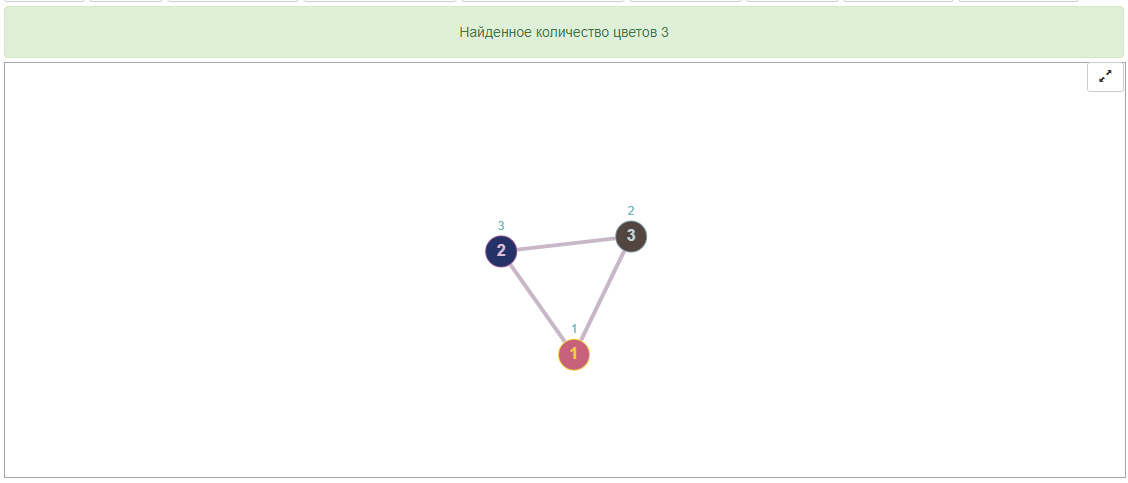
Входные данные:



Результат работы программы:

Ответ верный.



**Тест 2: (0,001sec)**

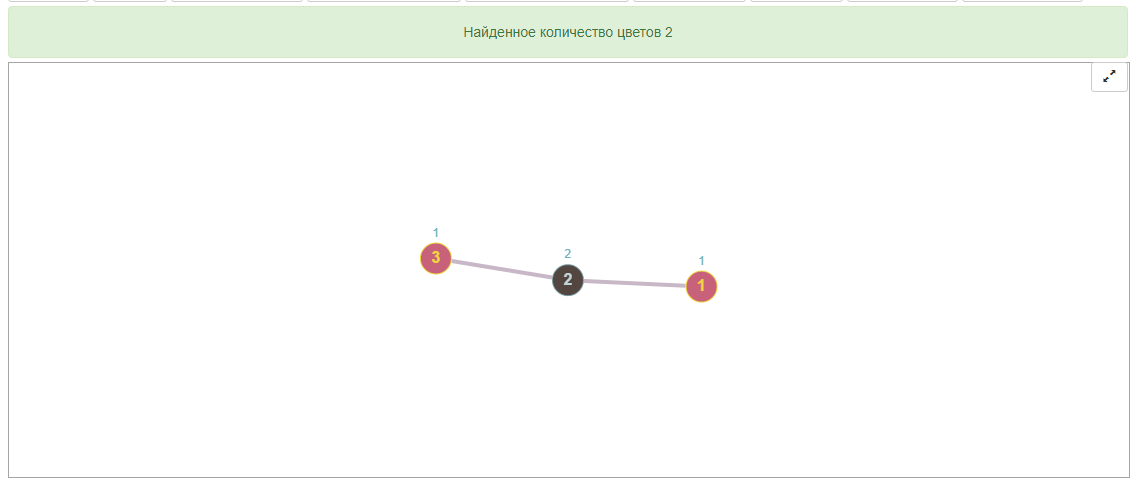
Входные данные:



Результат работы программы:

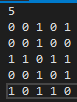
 

Ответ верный.



**Тест 3: (0,001sec)**

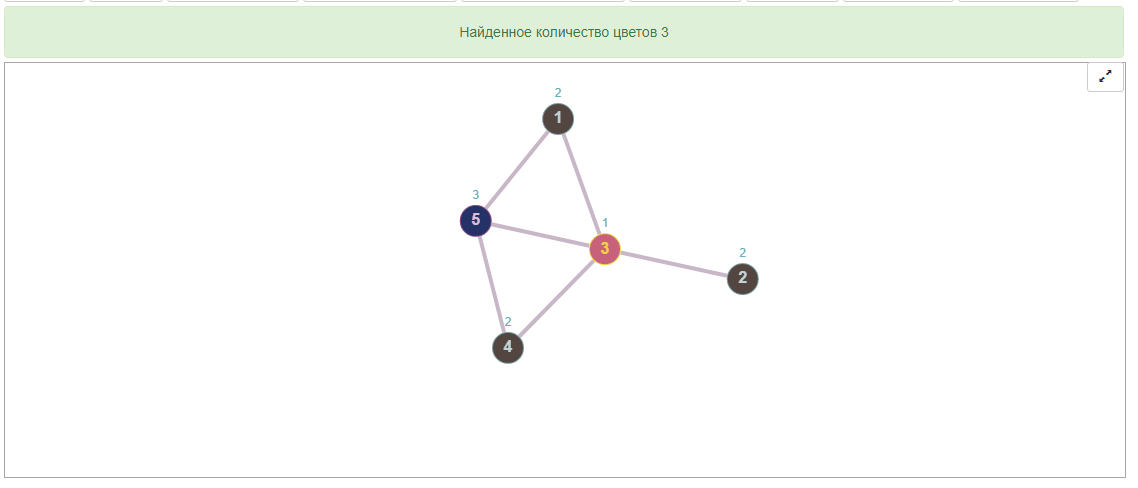
Входные данные:



Результат работы программы:

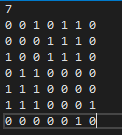
 

Ответ верный.



**Тест 4: (0,001sec)**

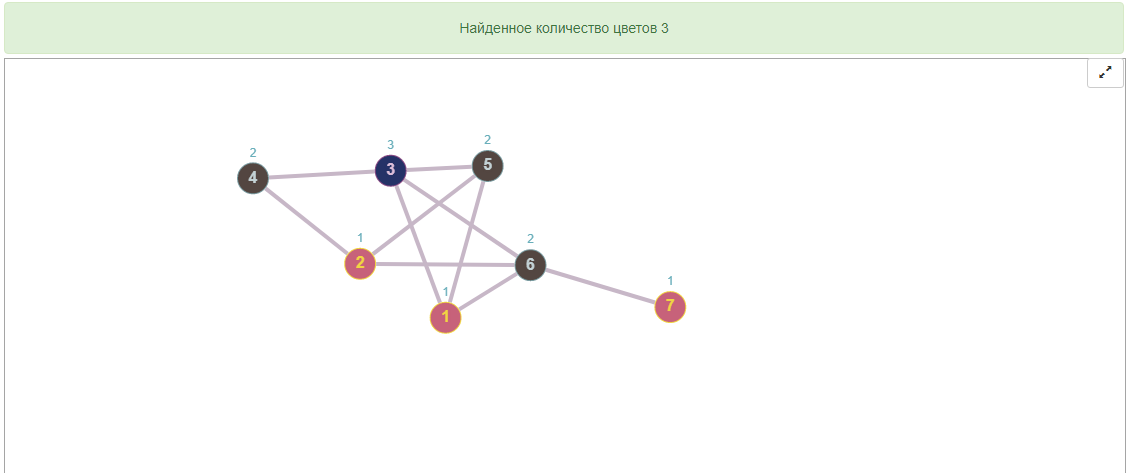
Входные данные:



Результат работы программы:

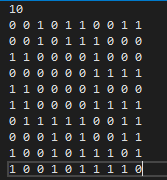
 

Ответ верный.



**Тест 5: (0,002sec)**

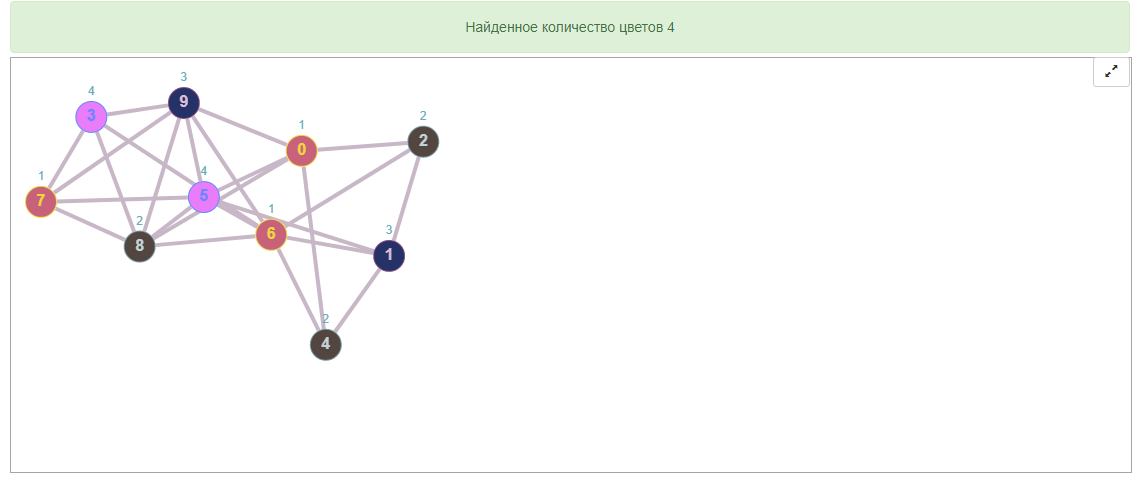
Входные данные:



Результат работы программы:

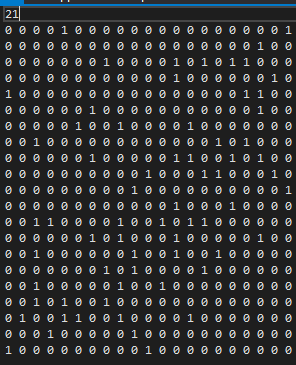
 

Ответ верный.



**Тест 6: (0,003sec)**

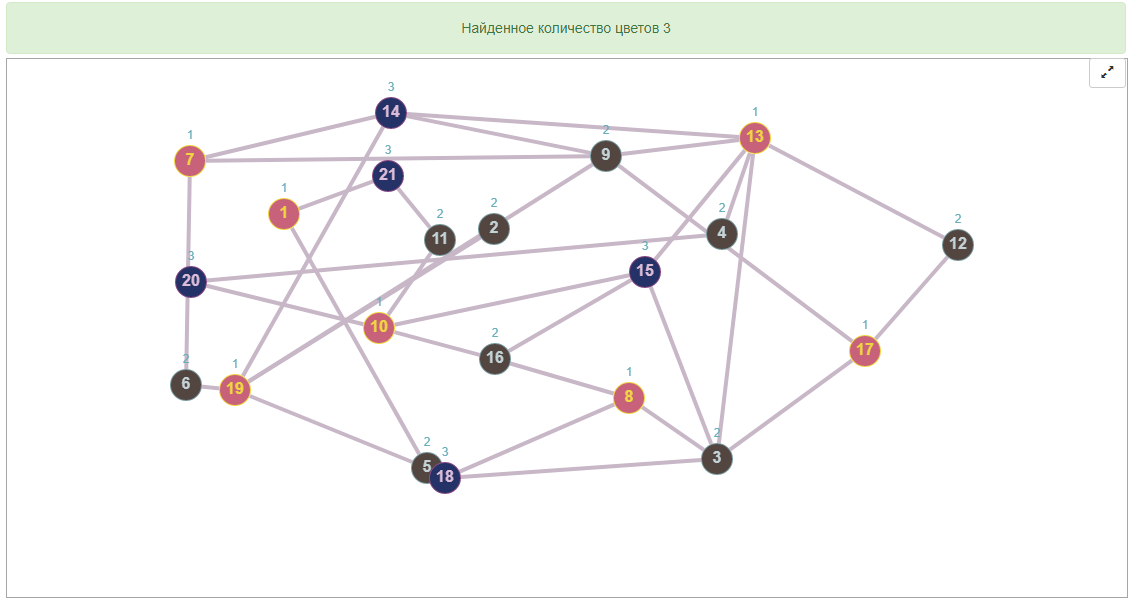
Входные данные:

****

Результат работы программы:

** **

Ответ верный.

****